

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-134446
(43)Date of publication of application : 18.05.2001

51)Int.Cl. G06F 9/45
G06F 15/16

21)Application number : 11-314130 (71)Applicant : NEC CORP
22)Date of filing : 04.11.1999 (72)Inventor : MURAI HITOSHI

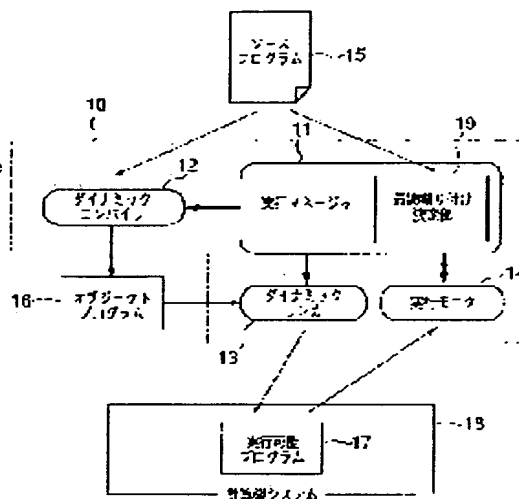
54) SYSTEM AND METHOD FOR OPTIMIZING MEMORY ALLOCATION, AND RECORDING MEDIUM

57)Abstract:

PROBLEM TO BE SOLVED: To reduce overheads at the of execution of such as suffix conversion, area dynamic allocation or the like while suppressing the necessary memory size of a program to the size of the memory of a computer system.

SOLUTION: Optimal memory allocation is decided by an optimal allocation deciding part 9 in the range of a memory size available at present by an execution time monitor 14.

A dynamic compiler 12 generates an object program for realizing the optimal memory allocation decided by the optimal allocation deciding part 19, and a dynamic linker 13 displaces one part of an object program without stopping the execution of an executable program.



LEGAL STATUS

Date of request for examination] 10.10.2000
Date of sending the examiner's decision of rejection] 08.09.2004
Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
Date of final disposal for application]
Patent number] 3633404
Date of registration] 07.01.2005
Number of appeal against examiner's decision of rejection] 2004-21018
Date of requesting appeal against examiner's decision of rejection] 08.10.2004
Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(11)特許出願公開番号

特開2001-134446

(P2001-134446A)

(43)公開日 平成13年5月18日(2001.5.18)

(51) Int.Cl.⁷

識別記号

FI

 δ -73-1* (参考)

G O 6 F 9/45

C O 6 F 15/16

630C 5B045

15/16

630

9/44

3 2 2 H 5 B 0 8 1

審査請求 有 請求項の数15 O.L (全 11 頁)

(21)出願番号

特願平11-314130

(22) 出題目

平成11年11月4日(1999.11.4)

(出願人による申告) 国等の委託研究の成果に係る特許出願(平成11年度通商産業省軽水炉改良技術確証試験等(発電設備診断システムの開発(学習・適応型情報処理による診断システムの開発)に関する委託研究、産業活力再生特別措置第30条の適用を受けるもの)

(71)出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72)発明者 村井 均

東京都港区芝五丁目7番1号 日本電気株式会社内

(74) 代理人 100071272

弁理士 後藤 洋介 (外1名)

Fターム(参考) 5B045 DD02 GG11

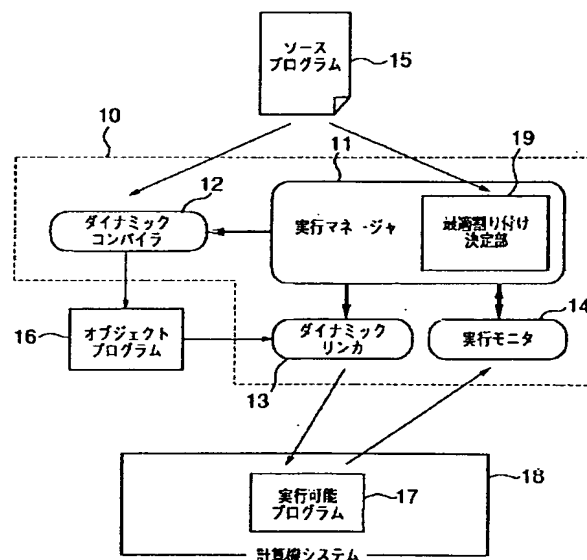
5B081 CC27 CC41

(54)【発明の名称】 メモリ割り付け最適化システム、方法、及び記録媒体

(57) 【要約】

【課題】 プログラムの必要メモリサイズを計算機システムの備えるメモリ以下に抑えつつ、添字変換や領域の動的割り付けといった実行時のオーバヘッドを削減する。

【解決手段】 実行時モニタ 14 が得た現在利用可能なメモリサイズの範囲内で、最適割り付け決定部 19 が最適なメモリ割り付けを決定する。ダイナミックコンパイラ 12 は、最適割り付け決定部 19 が決定した最適なメモリ割り付けを実現するオブジェクトプログラムを生成し、ダイナミックリンカ 13 は、実行可能プログラムの実行を停止することなくオブジェクトプログラムの一部を置換する。



【特許請求の範囲】

【請求項1】 プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化システムにおいて、

前記プロセッサエレメントで現在利用可能なメモリ サイズを取得する実行モニタと、

当該プログラムのソースプログラム、前記メモリ のサイズ、及び前記各データの重要度に基づいて、各データの最適なメモリ割り付け方法を決定する最適割り付け決定部とを有することを特徴とするメモリ割り付け最適化システム。

【請求項2】 プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化システムにおいて、

前記プロセッサエレメントで現在利用可能なメモリ のサイズを取得する実行モニタと、

当該プログラムのソースプログラム及び前記取得された前記メモリ のサイズに基づいて、前記各データの最適なメモリ割り付け方法を決定する最適割り付け決定部と、前記決定された各データの最適なメモリ割り付け方法に基づいて、前記ソースプログラムを変形し再コンパイルすることにより、オブジェクトプログラムを生成するダイナミックコンパイラと、

前記オブジェクトプログラムを、対応する元のオブジェクトプログラムと置換することにより、実行中の実行可能プログラムを変更するダイナミックリンカと、

前記各構成要素の動作を制御する実行マネージャとを有することを特徴とするメモリ割り付け最適化システム。

【請求項3】 請求項2に記載のメモリ割り付け最適化システムにおいて、

前記実行マネージャが、所定のタイミングで、前記各構成要素の動作を行うよう制御することを特徴とするメモリ割り付け最適化システム。

【請求項4】 請求項3に記載のメモリ割り付け最適化システムにおいて、

前記所定のタイミングが、当該並列計算機の所定の資源の利用状況が変化した時点であることを特徴とするメモリ割り付け最適化システム。

【請求項5】 請求項2に記載のメモリ割り付け最適化システムにおいて、

前記ダイナミックコンパイラが、メインプログラムで直接利用されるデータに関するメモリについても最適な割り付けを行うために、当該メインプログラムに相当する部分が、新たなメインプログラムから呼び出されるように、ソースプログラムを変形することを特徴とするメモリ割り付け最適化システム。

【請求項6】 請求項2に記載のメモリ割り付け最適化システムにおいて、

前記最適割り付け決定部での決定が、前記プログラム内の手続き呼び出し毎に行われ、割り付け方法が、前記手続き呼び出しの引数として指定された前記データ毎に決定されることを特徴とするメモリ割り付け最適化システム。

【請求項7】 請求項6に記載のメモリ割り付け最適化システムにおいて、

前記割り付け方法が、前記データのうち、当該プロセッサエレメントに関連する部分のみを前記プロセッサエレメントのメモリに割り付ける第1の方法と、前記データ全体を、当該プロセッサエレメントのメモリに割り付ける第2の方法とを含み、

前記最適割り付け決定部が、データの重要度を参照して、前記データを前記第1または第2の方法のいずれかの方法で割り付け、結果的に全てのデータが当該プロセッサエレメントのメモリに収容されるように、各データ毎の割り付け方法を決定することを特徴とするメモリ割り付け最適化システム。

【請求項8】 請求項7に記載のメモリ割り付け最適化システムにおいて、

前記最適割り付け決定部が、各プロセッサエレメント毎に、

(1) 前記第1の方法で割り付けられたデータのメモリ上のサイズの合計と前記第2の方法で割り付けられたデータのメモリ上のサイズの合計を加算した数が当該プロセッサのメモリの合計サイズ以下であり、

(2) 前記第2の方法で割り付けられたデータに対応する重要度が最大となるように、各データの割り付け方法を決定することを特徴とするメモリ割り付け最適化システム。

【請求項9】 請求項7に記載のメモリ割り付け最適化システムにおいて、

前記割り付け方法の決定が、NP問題の近似解をもとめる手法を用いて行われることを特徴とするメモリ割り付け最適化システム。

【請求項10】 請求項7に記載のメモリ割り付け最適化システムにおいて、

前記重要度が、前記プログラムによって、前記データがアクセスされる頻度であることを特徴とするメモリ割り付け最適化システム。

【請求項11】 請求項6に記載のメモリ割り付け最適化システムにおいて、

前記最適割り付け決定部で決定された割り付け方法が、1つのデータに対して複数存在する場合に、当該割り付け方法を1つに統一することを特徴とするメモリ割り付け最適化システム。

【請求項12】 プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プロ

グラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法において、

前記プロセッサエレメントで現在利用可能なメモリ サイズを取得するステップと、

当該プログラムのソースプログラム、前記メモリ のサイズ、及び前記各データの重要度に基づいて、各データ の最適なメモリ割り付け方法を決定するステップとを有す ることを特徴とするメモリ割り付け最適化方法。

【請求項13】 プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法において、

前記プロセッサエレメントで現在利用可能なメモリ のサイズを取得するステップと、

当該プログラムのソースプログラム及び前記取得された前記メモリ のサイズに基づいて、前記各データの最適なメモリ割り付け方法を決定するステップと、

前記決定された各データの最適なメモリ割り付け方法に基づいて、前記ソースプログラムを変形し再コンパイルすることにより、オブジェクトプログラムを生成するステップと、

前記オブジェクトプログラムを、対応する元のオブジェクトプログラムと置換することにより、実行中の実行可能プログラムを変更するステップとを有することを特徴とするメモリ割り付け最適化方法。

【請求項14】 プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法を実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記方法を実現させるプログラムは、

前記プロセッサエレメントで現在利用可能なメモリ のサイズを取得するステップと、

当該プログラムのソースプログラム、前記メモリ のサイズ、及び前記各データの重要度に基づいて、各データ の最適なメモリ割り付け方法を決定するステップとを有することを特徴とするプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項15】 プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法を実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記方法を実現させるプログラムは、

前記プロセッサエレメントで現在利用可能なメモリ のサイズを取得するステップと、

当該プログラムのソースプログラム及び前記取得された前記メモリ のサイズに基づいて、前記各データの最適なメモリ割り付け方法を決定するステップと、

前記決定された各データの最適なメモリ割り付け方法に基づいて、前記ソースプログラムを変形し再コンパイルすることにより、オブジェクトプログラムを生成するステップと、

前記オブジェクトプログラムを、対応する元のオブジェクトプログラムと置換することにより、実行中の実行可能プログラムを変更するステップとを有することを特徴とするプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、分散メモリ型並列計算機におけるメモリ割り付けの最適化方法に関し、特に、再コンパイルによって並列プログラムに対するメモリ割り付けを最適化するメモリ割り付け最適化方法に関する。

【0002】

【従来の技術】分散メモリ型並列計算機とは、各々がローカルメモリを持つ複数のプロセッサが、相互結合網（ネットワーク）で結合された構造の計算機システムである。通常、データは、各プロセッサ・エレメント（PE）のローカルメモリに分散して配置される。

【0003】逐次ソースプログラムから分散メモリ型並列計算機向けのオブジェクトプログラムを生成するコンパイラは、ソースプログラムに現れるデータ（配列）を、各PEのローカルメモリ上に分割・配置する。

【0004】これを実現する方式として、宣言された配列の領域（グローバル領域）のうち、各PEに分割・配置されたローカルな領域だけをローカルメモリに割り付ける方式がある（本明細書中、当該方式をSHRUNK方式と呼ぶことにする）。この方式では、ソースプログラムでは、グローバル領域上の添字式でアクセスされるよう記述されるのに対し、オブジェクトプログラムでは、ローカルメモリ領域上の添字式でアクセスする必要がある。従って、当該方式では、実行時にグローバル領域上の添字からローカル領域上の添字への変換処理が必要になる。

【0005】また、配列の分割を変更する際には、配列の領域を動的に割り付けたり、元の領域から値をコピーしたりする処理が必要になる。これらの処理は、プログラムの実行時間を増大させる要因となる。

【0006】尚、複数のPE間に亘って共有すべきデータがある場合、そのデータは、共有メモリや、データを使用する各PEのローカルメモリ等に配置される。

【0007】一方、配列を各プロセッサに配置する際に領域を分割せず、全プロセッサが配列の全体に等しい領域を保持し、そのうちのローカルな部分のみを用いると

いう方法も考えられる（ここでは、当該方式をNOSHRUNK方式と呼ぶことにする）。この方法は、HPF/JA（High Performance Fortran 2.0 公式マニュアル, High Performance Fortran Forum, シュプリング・フェアーク東京（1999））では、「フルシャドウ(FULL-SHADOW)」と呼ばれている。この方法では、必要な添字変換や領域の動的割り付けといった処理は不要であるため、実行時間を削減することができる。しかし、SHRUNK方式に比べると、メモリの使用効率が悪く、巨大な配列を扱うプログラムに適用することはできない。

【0008】配列に対してメモリ割り付け方法を個別に決めることができる場合、計算機システムのメモリサイズの範囲内でできるだけ多くの配列をNOSHRUNK方式で割り付ける方がよい。しかし、どの配列をNOSHRUNK方式で割り付けるのが最適であるかは、一般には利用可能なメモリサイズに依存し、利用可能なメモリサイズは種々に変化し得るため、最適なメモリ割り付け方法をコンパイル時に決定することはできない。

【0009】以下では、並列プログラムを記述するのに、HPF/JAの記法を用いる。

【0010】

【発明が解決しようとする課題】第1の問題点は、SHRUNK方式では、実行時に添字変換や領域の動的割り付けといったオーバーヘッドが発生することである。これは、ソースプログラムとオブジェクトプログラムでは配列のメモリ割り付け方法が異なることに起因する。

【0011】第2の問題点は、NOSHRUNK方式では、巨大な配列を扱えないことである。これは、当該方法では、配列全体をローカルメモリに配置しようとすることに起因する。

【0012】第3の問題点は、最適なメモリ割り付け方法をコンパイル時に決定することは一般にはできないことである。当該計算機システムの処理内容に応じて、システム的环境が変化すると、それに伴って、利用可能なメモリサイズも変化するからである。

【0013】従って、本発明の目的は、プログラムの必要メモリサイズを、現在利用可能なメモリサイズ以下に抑えつつ、以上の問題点を解決するメモリ割り付け最適化方法を提供することである。

【0014】

【課題を解決するための手段】本発明の第1の実施態様によれば、プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化システムにおいて、前記プロセッサエレメントで現在利用可能なメモリのサイズを取得する実行モニタと、当該プログラムのソースプログラム、前記メモリのサイズ、及び前記各データの重要度に基づいて、各データの最適なメモリ割り付け方法を決定する最適割り付け決定部とを有

するメモリ割り付け最適化システムが提供される。

【0015】また、本発明の第2の実施態様によれば、プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化システムにおいて、前記プロセッサエレメントで現在利用可能なメモリのサイズを取得する実行モニタと、当該プログラムのソースプログラム及び前記取得された前記メモリのサイズに基づいて、前記各データの最適なメモリ割り付け方法を決定する最適割り付け決定部と、前記決定された各データの最適なメモリ割り付け方法に基づいて、前記ソースプログラムを変形し再コンパイルすることにより、オブジェクトプログラムを生成するダイナミックコンパイラと、前記オブジェクトプログラムを、対応する元のオブジェクトプログラムと置換することにより、実行中の実行可能プログラムを変更するダイナミックリンカと、前記各構成要素の動作を制御する実行マネージャとを有するメモリ割り付け最適化システムが提供される。

【0016】また更に、本発明の第3の実施態様によれば、プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法において、前記プロセッサエレメントで現在利用可能なメモリのサイズを取得するステップと、当該プログラムのソースプログラム、前記メモリのサイズ、及び前記各データの重要度に基づいて、各データの最適なメモリ割り付け方法を決定するステップとを有するメモリ割り付け最適化方法が提供される。

【0017】また更に、本発明の第4の実施態様によれば、プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法において、前記プロセッサエレメントで現在利用可能なメモリのサイズを取得するステップと、当該プログラムのソースプログラム及び前記取得された前記メモリのサイズに基づいて、前記各データの最適なメモリ割り付け方法を決定するステップと、前記決定された各データの最適なメモリ割り付け方法に基づいて、前記ソースプログラムを変形し再コンパイルすることにより、オブジェクトプログラムを生成するステップと、前記オブジェクトプログラムを、対応する元のオブジェクトプログラムと置換することにより、実行中の実行可能プログラムを変更するステップとを有するメモリ割り付け最適化方法が提供される。

【0018】更に、本発明の第5の実施態様によれば、プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する

各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法を実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体が提供され、前記方法を実現させるプログラムは、前記プロセッサエレメントで現在利用可能なメモリのサイズを取得するステップと、当該プログラムのソースプログラム、前記メモリのサイズ、及び前記各データの重要度に基づいて、各データの最適なメモリ割り付け方法を決定するステップとを有する。

【0019】また更に、本発明の第6の実施態様によれば、プログラムが複数のプロセッサエレメントを備える並列計算機で実行される場合に、当該プログラムが使用する各データの前記プロセッサエレメントのメモリへの割り付けを最適化するメモリ割り付け最適化方法を実現させるプログラムを記録したコンピュータ読み取り可能な記録媒体が提供され、前記方法を実現させるプログラムは、前記プロセッサエレメントで現在利用可能なメモリのサイズを取得するステップと、当該プログラムのソースプログラム及び前記取得された前記メモリのサイズに基づいて、前記各データの最適なメモリ割り付け方法を決定するステップと、前記決定された各データの最適なメモリ割り付け方法に基づいて、前記ソースプログラムを変形し再コンパイルすることにより、オブジェクトプログラムを生成するステップと、前記オブジェクトプログラムを、対応する元のオブジェクトプログラムと置換することにより、実行中の実行可能プログラムを変更するステップとを有する。

【0020】

【発明の実施の形態】図1を参照して、本発明の第1の実施形態について図面を参照して詳細に説明する。

【0021】本発明のメモリ割り付け最適化システム10は、実行マネージャ11と、ダイナミックコンパイラ12と、ダイナミックリンカ13と、実行モニタ14とを含む。

【0022】実行マネージャ11は、最適割り付け決定部19を備えている。

【0023】実行マネージャ11は、ダイナミックコンパイラ12、ダイナミックリンカ13、実行モニタ14の制御を行う。

【0024】ダイナミックコンパイラ12は、実行マネージャ11より最適割り付け方法Ins-optを受け取って、ソースプログラム15を再コンパイルし、最適なメモリ割り付けを実現するオブジェクトプログラム16を

$$\sum_{i \in I_{ns}} m_i + \sum_{i \in I_{ns}} M_i \leq M_{all} \quad \dots (式1)$$

の元で、

$$W = \sum_{i \in I_{ns}} w_i \quad \dots (式2)$$

を最大にするような $I_{ns} (\subseteq I)$ を求める。ここで、式1の左辺の第1項は、ある1つのPEにおいて、実際に

生成する。ダイナミックコンパイラ12は、任意の配列がSHRUNK方式またはNOSHRUNK方式で割り付けられるようにソースプログラム15をコンパイルできる。

【0025】ダイナミックリンカ13は、実行可能プログラム17に含まれる手続きのうちダイナミックコンパイラ12の再コンパイル対象になったものを、新しいオブジェクトプログラム16で置換した後、リンクを行い、新しい実行可能プログラムを生成する。

【0026】実行モニタ14は、実行可能プログラム17や計算機システム18のオペレーティングシステムまたはハードウェアより、現在のメモリ割り付け方法Ins-current、現在利用可能なメモリのサイズMallを受け取り、実行マネージャ11に伝達する。

【0027】ここで、メモリ割り付け方法Insとは、NOSHRUNK方式で割り付けられる配列の集合である。また、利用可能なメモリとは、ある1つのPEが有するローカルメモリであって、実行可能プログラム17に関するスカラーデータや種々の実行時データが割り付けられているメモリ、あるいは計算機システム18上で動作する他の実行可能プログラムが使用中のメモリなどを除いたものである。

【0028】最適割り付け決定部19は、実行モニタ14より受け取った実行時情報とソースプログラム15から得られる情報を元に、計算機システム18における、ソースプログラム15に対する最適なメモリ割り付け方法を決定する。決定された方法は、ダイナミックコンパイラ12に伝達される。

【0029】ここで、最適なメモリ割り付け方法Ins-optの決定方法について説明する。

【0030】プログラム中で使われる配列A1, A2, ..., Anを、SHRUNK方式およびNOSHRUNK方式で割り付ける場合に必要となるメモリサイズをそれぞれ、m1, m2, ..., mnおよびM1, M2, ..., Mnとする。また、配列の重みをw1, w2, ..., wnとする。ここで、wiは、一定時間内に配列Aiがアクセスされた回数（アクセス頻度）とする。wiの値の設定に関しては、例えば、動的割り付けが生じる回数など他の基準も考えられるが、本発明の実施の形態はwiの値として何が設定されるかにはよらない。

【0031】このとき、最適割り付けを決定する問題は、次の式1、及び式2を用いて記述される。

【0032】（以下で、 $I = \{1, 2, \dots, n\}$ とする）。

【0033】

【数1】

【数2】

SHRUNK方式で割り当てられる配列のメモリサイズの合計であり、第2項は、当該PEにおいて、実際にNOSHRUNK

方式で割り当てられる配列のメモリサイズの合計を示す。式1は、実際に割り当てられるメモリサイズの合計が、1つのPEの使用可能ローカルメモリの合計サイズを越えないことを示している。

【0034】このような条件は、一般には、当該計算機システム内のPE全てを考慮して設定すべきであるが、ここでは、問題を単純化するため、ある1つのPEについて考える。このとき、他のPEでも前記1つのPEと同一の動作（即ち、同一のメモリ割り当て）が行われていると仮定する。

【0035】ここで、

$$\sum_{i \in \text{Ins}} M'_{i} \leq M_{\text{all}} - \sum_{i \in I} m_i = M'_{\text{all}} \quad \dots (式4)$$

の元で、

$$W = \sum_{i \in \text{Ins}} w_i \quad \dots (式5)$$

を最大にするような $\text{Ins} (\subseteq I)$ を求める。

【0037】これは、いわゆるナップザック問題として把握することが可能である。ナップザック問題はNP問題であることが知られており、配列の数 n に対して多項式時間内で解くことはできない。しかし、多項式時間内で近似解を得る方法は多く知られており（Sahni, Sarta j K., "Approximate algorithms for the 0/1 knapsack problem, "Journal of the ACM, vol. 22, no. 1, pp. 115-124, January, 1975を参照されたい）、最適割り付け決定部19は、そのうちの任意の1つの方法で Ins-opt の近似解を求めることができる。

【0038】次に、図1および図2を参照して本実施の形態の動作について詳細に説明する。

【0039】実行モニタ14は、実行可能プログラム17や計算機システム18のオペレーティングシステムまたはハードウェアより、現在のメモリ割り付け方法 Ins-current 、現在利用可能なメモリサイズ M_{all} を受け取り、実行マネージャ11に伝達する（図2のステップ21）。

【0040】最適割り付け決定部19は、現在利用可能なメモリサイズ M_{all} およびソースプログラムの配列出現情報から、最適なメモリ割り付け方法 Ins-opt を求める（ステップ22）。

【0041】実行マネージャ11は、実行モニタ14より得た Ins-current が Ins-opt と等しいか否かを判定する（ステップ23）。

【0042】 Ins-current が Ins-opt と異なる場合、実行マネージャ11は、ダイナミックコンパイラ12を起動し、 Ins-opt を伝達する。

【0043】ダイナミックコンパイラ12は、配列 A_i (i は Ins-opt に含まれる) が NOSHRUNK 方式で、配列 A_j (j は Ins-opt に含まれない) が SHRUNK 方式でそれぞれ割り付けられるようにソースプログラム15をコンパイルし、新しいオブジェクトプログラム16を生成する（ス

$$M'_{i} \equiv M_i - m_i \quad \dots (式3)$$

とすると、この問題は次の式4、及び式5を用いて書き換えられる。但し、上記式3における M_i と m_i はそれぞれ、配列 A_i を NOSHRUNK 方式で割り当てるとした場合のメモリサイズ、配列 A_i を SHRUNK 方式で割り当てるとした場合のメモリサイズを示しており、これらが実際に同時に割り当てられる訳ではない。式1に示すように、ある配列 A_i のローカルメモリへの割り当てに関しては、 NOSHRUNK 方式か SHRUNK 方式のどちらかが用いられる。

【0036】

【数3】

【数4】

テップ24）。

【0044】本実施形態は、プログラムの実行の間に、対象の手続きが複数回実行される場合に大きな効果がある。従って、例えば、図3のプログラムのように、ループ中から呼ばれて複数回実行される手続きを対象とすることが好ましい。

【0045】ここで、メインプログラムは、ダイナミックコンパイラ12の再コンパイル処理の対象にはできない。このような制限が存在するのは、ダイナミックコンパイラ12における再コンパイルが、メインプログラム内の手続き呼び出しを対象として行われるためである。

【0046】また、メモリ割り付け最適化の対象となるのは、ダイナミックコンパイラ12の再コンパイル処理の対象手続き内でのみ割り付けが行われる配列である。

【0047】ダイナミックリンカ13は、まず実行可能プログラム17の動作を一時的に中断させる。次に、実行可能プログラム17に含まれる手続きのうちダイナミックコンパイラの再コンパイル対象になったものを、新しいオブジェクトプログラム16で置換する。次に、それらのオブジェクトプログラムをリンクして新しい実行可能プログラムを生成し、実行可能プログラムの動作を再開させる（ステップ25）。

【0048】ステップ23において、 Ins-current が Ins-opt と等しい場合、本発明の実施の形態は何もせずに処理を終了する。

【0049】次に、具体的な実施例を用いて本実施の形態の動作を説明する。

【0050】配列 A_1 ないし A_8 の現れる、図3のようなプログラムを考える。この場合の、 m_i 、 M_i 、 M'_{i} 、 w_i のそれぞれの値が、図4の表に示されている。

【0051】ここで、 A_1 はメインプログラムで割り付けられるので、上述の通り、本実施形態における最適化の対象にはならない。

【0052】また、 $M_{\text{all}} = 1000$ であるとする。

【0053】図4を参照すると、全ての配列がSHRUNK方式で割り付けられた場合には、

$$M'all = Mall - \sum mi = 1000 - 160 = 840$$

となる。

【0054】ここで、全ての配列がSHRUNK方式で割り付けられるようにコンパイルされた実行可能プログラムが、計算機システム上で動作しているものとする。このとき、

$$Mcurrent = 160$$

$$Ins-current = \{\}$$

である。

【0055】ステップ21において、実行モニタは、計算機システムおよび実行可能プログラムから、上記の Mall, Mcurrent, Ins-current を得て、実行マネージャへ伝達する。

【0056】ステップ22において、実行マネージャは最適割り付け決定部を起動する。

【0057】最適割り付け決定部は、実行マネージャより伝達された Mall とソースプログラムから抽出した配列情報から、最適割り付け方法 Ins-opt を計算する。

【0058】ここでは、wiの値の大きい配列から順に Ins-opt に加えていく「greedy アルゴリズム」を用いる。その結果、

$$Ins-opt = \{3, 6, 7, 8\}$$

が得られる。

【0059】ステップ23において、Ins-current と Ins-opt は異なるので、ステップ24に移る。

【0060】ステップ24において、実行マネージャはダイナミックコンパイラを起動する。ダイナミックコンパイラは、Ins-opt に含まれる配列A3, A6, A7, A8がNO SHRUNK方式で、その他の配列がSHRUNK方式で割り付けられるようにソースプログラムを再コンパイルし、新しいオブジェクトプログラムを生成する。

【0061】ステップ25において、ダイナミックリンカは、新しいオブジェクトプログラムと、実行可能プログラムに含まれるオブジェクトプログラムを置換する。

【0062】その結果、

$$Mcurrent = 916 \leq Mall$$

となり、Mall の範囲内で、4つの配列A3, A6, A7, A8をNOSHRUNK方式で割り付けることができる。

【0063】次に、本発明の第2の実施形態の動作について説明する。

【0064】上述した第1の実施形態においては、実行モニタ14及び実行マネージャ11は、任意のタイミングで1度だけ実行され得る。これに対し、第2の実施形態では、所定のタイミングで実行マネージャを起動することにより、第1の実施形態で行われた処理を繰り返す。実行マネージャを起動するタイミングとしては、例えば、計算機システム上で動作する他の実行可能プログラムが終了した時点や配列の分割・配置方法が変更さ

れた時点などが考えられる。

【0065】本実施形態では、プログラムの実行中に計算機システムの状態が変化した場合であっても、それに応じてメモリ割り付け方法を動的に変化させることができる。

【0066】次に、本発明の第3の実施形態の動作について説明する。

【0067】第1の実施形態では、メインプログラムを再コンパイルの対象にできなかった。即ち、メインプログラムで割り付けが行われる配列に対しては割り付け方法を最適化できなかった。そこで、本発明の第3の実施形態では、ダイナミックコンパイラ12がダミーのメインプログラムを生成し、本来のメインプログラムはこのダミーのメインプログラムから呼び出されるようにソースプログラムを変形する。この際に、本来の再コンパイル対象である手続き呼び出しを囲むループにストリップマイニングを適用し、その外側ループはダミーのメインプログラムに置く。また、本来のメインプログラム内で宣言される変数は共通ブロック実体とする。その後、本来のメインプログラムであった手続きを再コンパイルの対象として、第1の実施形態を適用する。

【0068】図3のソースプログラムに本実施形態を適用すると、ダイナミックコンパイラ12は、図5のプログラムを生成する。この場合のmi, Mi, M'i, wi のそれぞれの値は、図6の表に示すようなものである。

【0069】また、Mall = 1000とする。

【0070】従って、図6の表から、 $M'all = Mall - \sum mi = 1000 - 200 = 800$ が得られる。

【0071】ここで、第1の実施形態を適用すれば、

$$Ins-opt = \{1, 3, 8\}$$

が得られる。

【0072】本実施形態では、上述のように、メインプログラムも再コンパイルの対象にすることができるため、メインプログラムで割り付けが行われる配列に対しても本発明を適用できる。

【0073】次に、本発明の第4の実施形態について詳細に説明する。

【0074】前述の通り、第1の実施形態では、再コンパイル対象の各手続きに対して個別に最適割り付け方法を決定する。従って、ある手続き呼び出しに関して、実引数と仮引数で異なる割り付け方法が選択される可能性があり、その場合、実行時に割り付け方法を変換するオーバーヘッドが生じる。

【0075】例えば、

CALL SUB1 (A1, A2)

CALL SUB2 (A2, A3)

というコードが、再コンパイルされるプログラムのソースコードに記述されている場合、本発明の第1の実施形態では、手続き単位に、各配列に対してSHRUNK方式かNO SHRUNK方式が決定されるため、上記SUB1の引数に示さ

れた配列A2が、SHRUNK方式と決定され、上記SUB 2の引数に示された配列A2がNOSHRUNK方式と決定される場合がある。実際には、1つの方法で配列A2が配置されるため、当初決定された方式で配置されなかったことになる手続き、SUB 1、SUB 2のどちらかは、配列A2をアクセスする際に、その異なる方式用に、アドレス等の変換を行う必要が生じる。

【0076】そこで、第4の実施形態では、最適割り付け決定部は、手続き呼び出しの実引数と仮引数の割り付け方法をできるだけ一致させるように各配列の割り付け方法を決定する。具体的には、決定された1つの配列の割り付け方式が、手続き間で異なっている場合に、当該割り付け方法を、ローカルメモリの合計を越えないという条件の下で、どちらかの方法に統一し、その統一された方法に基づいてオブジェクトプログラムを生成（又は再生成）する。

【0077】本実施形態により、割り付け方法を変換するための実行時オーバーヘッドを削減することができる。

【0078】次に、図7を参照して、本発明の各実施形態が実行される分散メモリ型並列計算機システムの一構成例について説明する。尚、図1に示した構成要素と同一の要素については、同一の符号が付されている。

【0079】当該計算機システム18は、 n 個のプロセッサエレメントPE1ないしPE n （31₁～31 _{n} ）を有している。PEの数は、例えば4、8、16といった数である。これら複数のPEは、ネットワーク32によって相互に接続されている。また、各PEは、それぞれ、ローカルメモリ及びプロセッサを有している。各プロセッサは、対応するローカルメモリ（又は共有メモリ）にロードされたプログラムに従って、ローカルメモリ内のデータその他に対して並列処理を実行する。

【0080】本発明によるメモリ割り付け最適化システム10は、ネットワーク32に接続され、そのネットワークを介して、各PEに動的に再コンパイルされたプログラムを送信し、データの割り付けの変更等を指示する。また、メモリ割り付け最適化システム10内の実行モニタ14は、当該ネットワーク32を介して、各PEのローカルメモリの空き容量の状況等を取得する。

【0081】この例では、メモリ割り付け最適化システム10は、PEとは異なる、例えばホストマシン内で実行されるものとして示されているが、1つのPEを使用して当該システム10を実現することも可能である。

【0082】記録媒体33は、ソースプログラム15や、メモリ割り付け最適化方法を実現するためのプログラム自体を含み、必要に応じてメモリ割り付け最適化システム10で読み取られる。こうした記録媒体33の代表的な例は、フロッピー（登録商標）ディスクやCD-ROMである。

【0083】

【発明の効果】以上に説明したように、本発明では、以下の効果を得ることが出来る。

【0084】第1の効果は、頻繁にアクセスされる配列を、重要度に応じて、できるだけオーバーヘッドの小さいNOSHRUNK方式で割り付けることにより、添字変換や動的割り付けといったオーバーヘッドを削減できることである。

【0085】第2の効果は、必要メモリの小さいSHRUNK方式で割り付けることにより、巨大な配列を扱えることである。

【0086】第3の効果は、実行時に、計算機システムの現在の状態に応じて最適なメモリ割り付け方式を決定し、それに従って生成されたオブジェクトプログラムで、実行中の実行可能プログラムの一部を（実行を停止することなく）置換するため、計算機システムの状態に応じて、常に最適なメモリ割り付け方式を達成できることである。

【図面の簡単な説明】

【図1】本発明の第1、第2、第3および第4の実施形態の構成を示すブロック図である。

【図2】本発明の第1の実施形態の動作を示す流れ図である。

【図3】本発明の第1の実施形態において入力されるソースプログラムのコードの一部を示す図である。

【図4】各配列毎に、SHRUNK方式、NOSHRUNK方式によるメモリ割り当てサイズ等を示す表である。

【図5】本発明の第3の実施形態において入力されるソースプログラムのコードの一部を示す図である。

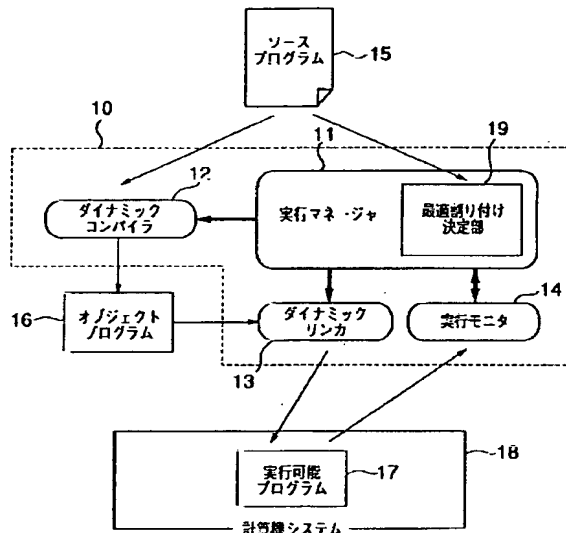
【図6】各配列毎に、SHRUNK方式、NOSHRUNK方式によるメモリ割り当てサイズ等を示す表である。

【図7】本発明が実行される分散メモリ並列処理計算機システムの一構成例を示すブロック図である。

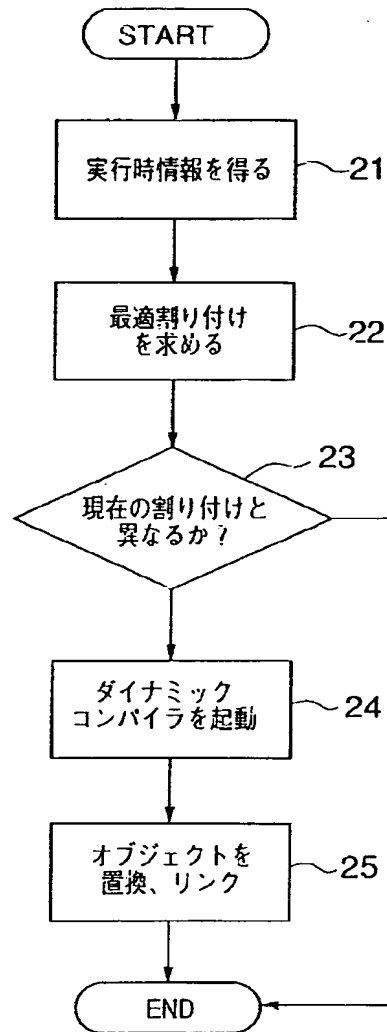
【符号の説明】

- 10 メモリ割り付け最適化システム
- 11 実行マネージャ
- 12 ダイナミックコンパイラ
- 13 ダイナミックリンカ
- 14 実行モニタ
- 15 ソースプログラム
- 16 オブジェクトプログラム
- 17 実行可能プログラム
- 18 計算機システム
- 19 最適割り付け決定部
- 31₁～31 _{n} プロセッサエレメント（PE）
- 32 ネットワーク
- 33 記録媒体

【図1】



【図2】



【図3】

```

PROGRAM SAMPLE
REAL A1 (100)
!HPF$ PROCESSORS P (10)
!HPF$ DISTRIBUTE (BLOCK) ONTO P:: A1
DO I=1,100
  ...
  CALL SUB (A1)
  ...
END DO
END

SUBROUTINE SUB (A1)
  REAL A1 (100)
  REAL A2 (100), A3 (200), A4 (40), A5 (50),
    A6 (80), A7 (20), A8 (10)
  ...
  ...
END
  
```

【図4】

	mi	Mi	M' i	wi
A2	40	400	380	100
A3	40	400	380	240
A4	16	180	144	80
A5	20	200	180	80
A6	32	320	288	150
A7	8	80	72	250
A8	4	40	38	20

【図6】

	mi	Mi	M' i	wi
A1	40	400	380	200
A2	40	400	380	100
A3	40	400	380	240
A4	16	180	144	80
A5	20	200	180	80
A6	32	320	288	150
A7	8	80	72	250
A8	4	40	38	20

(表 0) 101-134446 (P2001-134446A)

【図 5】

```
PROGRAM DUMMY_MAIN
DO I=1,10
  CALL SAMPLE
END DO

END

SUBROUTINE SAMPLE

REAL A1 (100)
COMMON A1

!HPF$ PROCESSORS P (10)
!HPF$ DISTRIBUTE (BLOCK) ONTO P:: A1

DO I=1,10
  ...
  CALL SUB (A1)
  ...
END DO

END

SUBROUTINE SUB (A1)

REAL A1 (100)
REAL A2 (100), A3 (200), A4 (40), A5 (50),
\  A6 (80), A7 (20), A8 (10)

!HPF$ PROCESSORS P(10)
!HPF$ DISTRIBUTE (BLOCK) ONTO P:: A1
!HPF$ DISTRIBUTE (BLOCK) ONTO P:: A2, A3, A4, A5,
!HPF$ A6, A7, A8

...

END
```

(特1) 01-134446 (P2001-134446A)

【図7】

